1

The anime dataset from https://www.kaggle.com/datasets/andreuvallhernndez/myanimelist was used in this project. The rows represent animes.

One approach for this dataset would be to find a simple (i.e. 1 degree of freedom) relationship between each pair of continuous variables. This can help identify some simple relationships in the data. A simple linear regression would suffice if both variables are not significantly skewed. However, it is very unlikely that this is always the case. For example, it can be assumed that variables involving the popularity of an anime (i.e. members, scored_by, and favorites) are highly skewed to the left, since for animes (and many things in general), more popular ones are rarer; this is analogous to the wealth pyramid and the caste system.

When a variable is skewed, a box-cox transformation can be used to fix this and also improve the model fit before applying the linear regression. A simple way to do this would be to apply a power (or log) from Tukey's ladder of powers on that variable based on the skewness (i.e. lower power for more left skewness). However, if one wants to be more precise, then for each pair of variables (x, y), one can choose the lambda that minimizes the adjusted MSE $(\lambda * GM(y))^{-2} * MSE(\hat{\beta})$ (where GM is the geometric mean and $\hat{\beta}$ is the estimate of the coefficient for the linear model $y^{\lambda} \sim x$) and then apply the power of lambda (or log, if lambda=0) to to y; this procedure can then be repeated for x.

Another approach for this dataset would be to determine if, for a categorical variable, an observation in one class is different from that of another class based on other variables. For categorical variables with a low number of classes (such as status, start_season, sfw, approved), a Bonferonni test can be used. However, in general, Tukey's multiple comparison test should be used. This can be done by using the TukeyHSD function in R. When passing an anova table for y vs x, then for each pair of class in y, the 95% family-wise confidence intervals and p-values are determined, where a p-value$<$0.05 indicates a significant difference in x between those 2 classes.

2

The code for cleaning data is shown in the next questions. It involves converting any strings representing dates and/or times into Date or numeric values. For simplicity, no imputation was used; I got rid of NA values by simply removing rows and columns. For question 6, categorical variables with either 1 or too many classes (i.e. "approved" and last 14 variables), id variables (i.e. first 2 columns), and too many NA values (i.e. broadcast_time) were dropped. Still, after dropping rows with NA's, more than half of the observations remained.

3

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.3     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.0.0 --
## v broom        1.0.0     v rsample      1.1.1
## v dials        1.1.0     v tune         1.0.1
```

```
## v infer        1.0.4     v workflows    1.1.3
## v modeldata    1.1.0     v workflowsets 1.0.0
## v parsnip      1.0.4     v yardstick    1.1.0
## v recipes      1.0.5
## -- Conflicts ---------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Use tidymodels_prefer() to resolve common conflicts.
```

```
library(skimr)
library(stringr)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
dat0=read.csv("anime.csv")
#skim(dat0)
dat1=dat0[c("type","score","source","members","start_date","total_duration")]
dat1$start_date=as.Date(dat1$start_date)
as.Time.numeric = function(S,days=T) {
  sapply(S,function(s){
    if (days) {
      s1=str_split(s,"days")[[1]]
    }
    else {
      s1=c(0,s)
    }
    s2=str_split(s1[2],":")[[1]]
    as.numeric(s1[1])*86400+as.numeric(s2[1])*3600+as.numeric(s2[2])*60+as.numeric(s2[3])
  })
}
dat1$total_duration=as.Time.numeric(dat1$total_duration)
dat=na.omit(dat1)
count(dat,source) #ggpairs doesn't accept variables with >15 classes, so remove 2 with the lowest count
```

```
##          source    n
## 1               1750
## 2  4_koma_manga  264
## 3          book   81
## 4     card_game   59
## 5          game  824
## 6   light_novel  807
## 7         manga 3838
## 8   mixed_media   54
## 9         music  268
## 10        novel  487
## 11     original 3911
## 12        other  432
## 13 picture_book   46
```

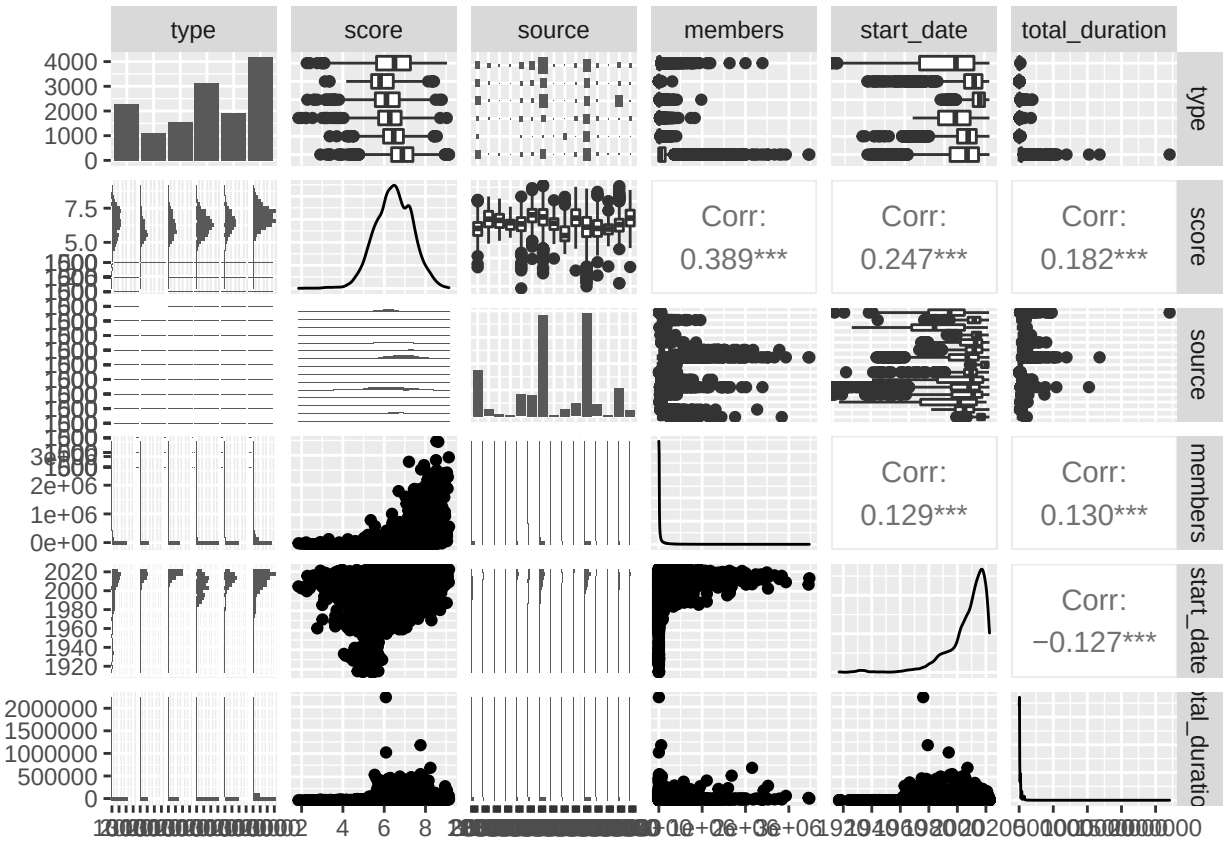
```
## 14        radio    7
## 15 visual_novel 1063
## 16    web_manga  248
## 17    web_novel   13
```

```
dat$source[which(dat$source=="radio" | dat$source=="web_novel")]="other"
count(dat,type) #only 1 with type=""; remove
```

```
##      type    n
## 1            1
## 2   movie 2279
## 3   music 1109
## 4     ona 1540
## 5     ova 3143
## 6 special 1906
## 7      tv 4174
```

```
dat=dat[-which(dat$type==""),]
n=nrow(dat)
dat$type=as.factor(dat$type)
dat$source=as.factor(dat$source)
ggpairs(dat)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The selected variables were: type, score, source, members, start_date, and total_duration. I believe these are the most important variables since they are the most different measures of anime (unlike for example, start_date, start_year, created_at, and real_start_date, which are all measures of when an anime started, so including these would decrease the importance of each other).

It appears that the continuous variables (i.e. score, members, start_date, and total_duration) have a positive trend with each other, although they appear to be nonlinear (a strictly convex function appears to fit members vs score better than a straight line, while a strictly concave function appears to fit start_date vs members better than a straight line), heteroscedastic (for example, members, start_date, and total_duration varies more as members increases, is between around 4-6, and is between around 5-9, respectively), and/or non-normal (members and total_duration are highly skewed left, and therefore so would the plots where either one is the x).

Due to the extreme left skewness of members and total_duration, only a few observations with top values in those variables (i.e. the right tails of the distribution) for classes of type and source can be seen. From these values, it appears that type=tv has significantly more members and total duration than that of other classes. Also, it appears that the scores for each class of type are roughly similar (possibly except that type=tv has higher score than movie), but different for each class of source, as many of the corresponding box and whisker plots do not overlap. Similarly, different classes of source or type have different values of start_date.

The shapes of the box and whisker plots involving score appear to have roughly similar size and shape and are roughly symmetric, which indicates similar values of variance and kurtosis and no significant skewness in score for different classes in type and source. However, this is not the case for start_date. The left whiskers are longer than the right, which indicates right skewness; also, for example, for type=movie, the range of points on the left is much smaller than the left whisker, while it is the opposite case for type=music, which indicates that the kurtosis of start_date is higher for type=movie than for type=music.

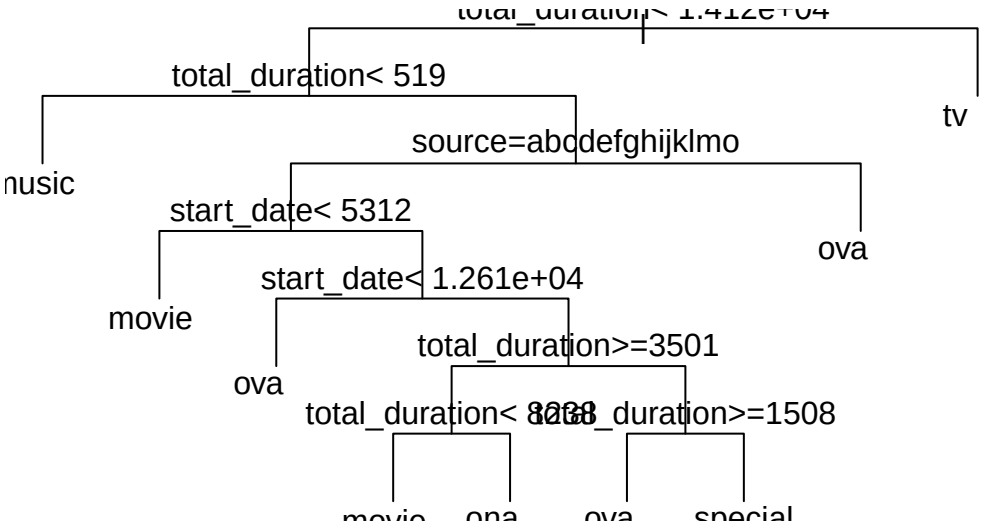
4

```
library(rpart)
```

```
##
## Attaching package: 'rpart'
```

```
## The following object is masked from 'package:dials':
##
##     prune
```

```
M=rpart(type~.,data=dat)
plot(M,uniform=T)
text(M)
```




```
#print(M)
pred=predict(M,dat)
typenames=sort(as.character(unique(dat$type)))
pred_class=sapply(1:n,function(i) {
  typenames[which.max(pred[i,])]
})
confusion=dat
confusion$pred_class=factor(pred_class,levels=typenames)
conf_mat(confusion,type,pred_class) #confusion matrix
```

```
##           Truth
## Prediction movie music  ona  ova special   tv
##    movie    1072     1  125  241     162  133
##    music     398  1065  592   65     546    8
##    ona        12     0  114   38      12  104
##    ova       664    17  217 2440     553  264
##    special   131    26  231  278     628   60
##    tv          2     0  261   81       5 3605
```

```
as.Date(5312,origin="1970-01-01")
```

```
## [1] "1984-07-18"
```

```
as.Date(12611,origin="1970-01-01")
```

```
## [1] "2004-07-12"
```

```
predict(M,data.frame(score=8.66,source="manga",members=1922827,
                     start_date=as.Date("1999-10-20"),total_duration=1527752))
```

```
##          movie music       ona        ova     special       tv
## 1 0.0005058169     0 0.0660091 0.02048558 0.001264542 0.911735
```

```
predict(M,data.frame(score=8.34,source="other",members=46495,
                     start_date=as.Date("2019-05-05"),total_duration=12060))
```

```
##        movie music       ona       ova    special        tv
## 1 0.04285714     0 0.4071429 0.1357143 0.04285714 0.3714286
```

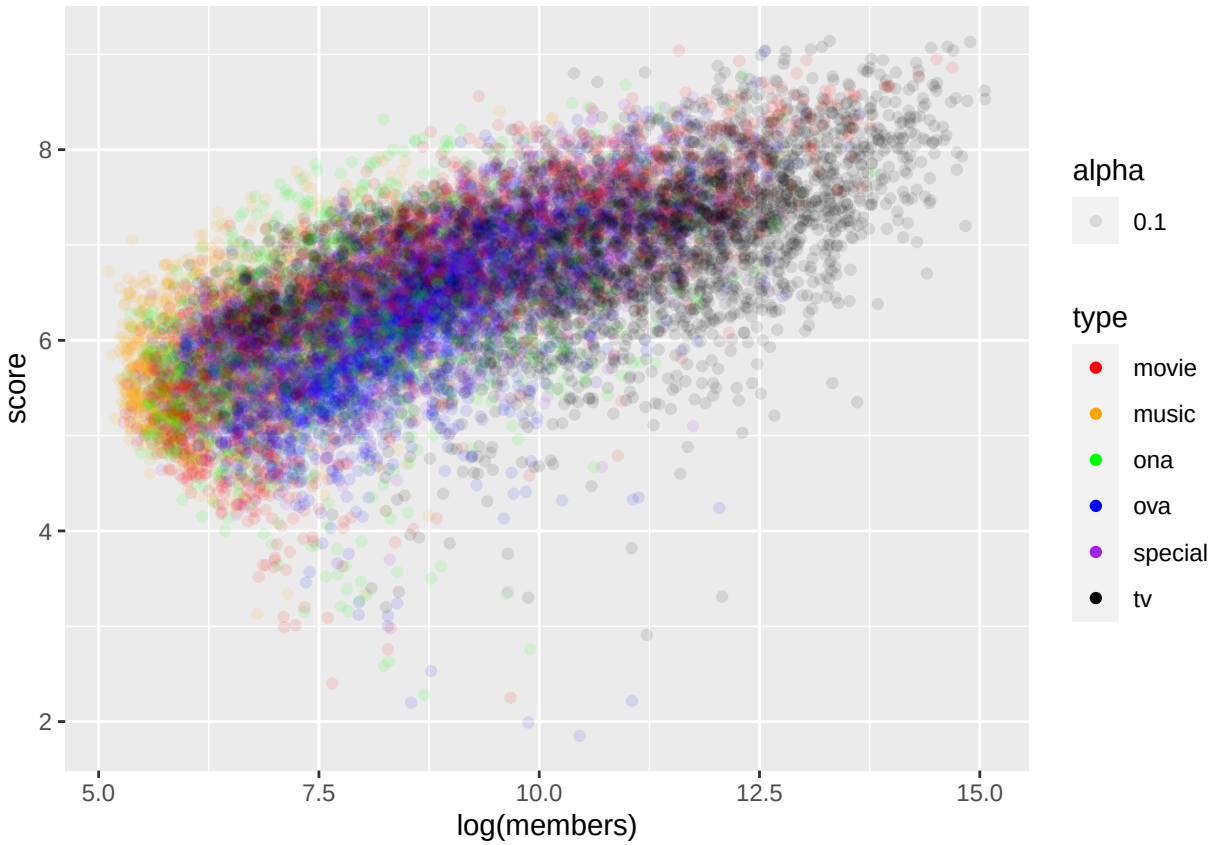
The root node and the following node are both split by total_duration, since they maximizes its information gain in the corresponding nodes. (This is reasonable because, from personal experience, usually anime of tv types are the longest, while music types are the shortest; indeed, we see that total_duration>=14117.5 predicts type=tv and total_duration<519 predicts type=music). The rest of the tree follows similarly, with nodes being split by the variable that yields the highest information gain. Information gain is measured by decrease in entropy, so splits that result in child nodes having higher proportion of training data in one class would have a higher information gain.

Example prediction 1: score=8.66, source="manga", members=1922827, start_date="1999-10-20", total_duration=1527752. Starting at the root, the total_duration is 1527752>=14117.5, so move to right child, which predicts type=tv.

Example prediction 2: score=8.34, source="other", members=46495, start_date="2019-05-05", total_duration=12060, Starting at the root, the total_duration is 12060<14117.5, so move to left child. The total_duration is 12060>=519, so move to right child. The source is "other" which starts with "o", so move to left child. The start_date is "2019-05-05">="1984-07-18", so move to right child. The start_date is "2019-05-05">="2004-07-12", so move to right child. The total_duration is 12060>=3501, so move to left child. The total_duration is 12060>=8238, so move to right child, which predicts type=ona.

5

```
ggplot(dat)+aes(log(members),score)+geom_point(aes(color=type,alpha=0.1))+
  scale_color_manual(values=c("movie"="red","music"="orange","ona"="green",
                              "ova"="blue","special"="purple","tv"="black"))+
  scale_alpha(range=0.1)
```



From the plot, it appears that score is roughly increasing linearly with log members. Also, it appears that music, ona, and ova types tends to have the lowest scores, with music also having the least members, and the distributions of ona and ova on the plot have similar location but ona has more spread. Finally, tv types tend to have the highest scores and members, and its distribution appears to have the most spread.

6

```
dat2=dat0[3:24]
dat2$episode_duration=as.Time.numeric(dat2$episode_duration)
dat2$total_duration=as.Time.numeric(dat2$total_duration)
dat2$start_date=as.Date(dat2$start_date)
dat2$end_date=as.Date(dat2$end_date)
dat2$created_at=as.Date(dat2$created_at)
dat2$updated_at=as.Date(dat2$updated_at)
dat2$real_start_date=as.Date(dat2$real_start_date)
dat2$real_end_date=as.Date(dat2$real_end_date)
dat2=na.omit(dat2)
unique(dat2$approved) #only 1 value, so remove
```

```
## [1] "True"
```

```
dat2=dat2[-which(colnames(dat2)=="approved")]
library(glmnet)
```

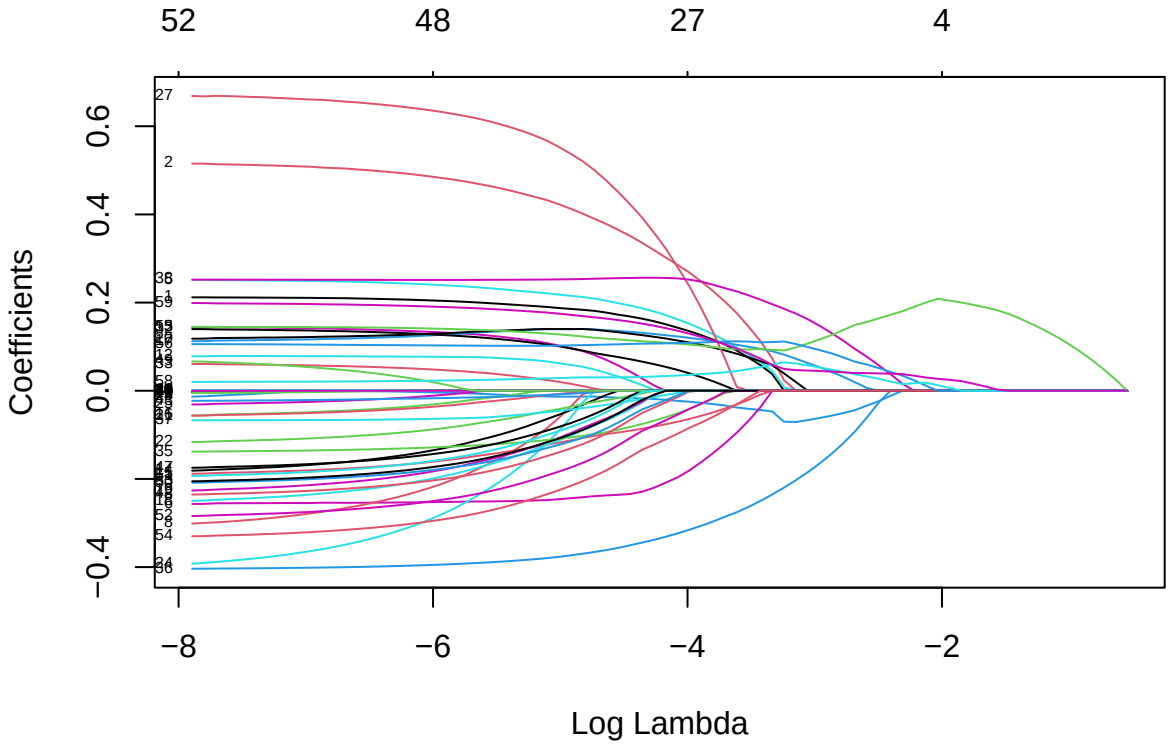
```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

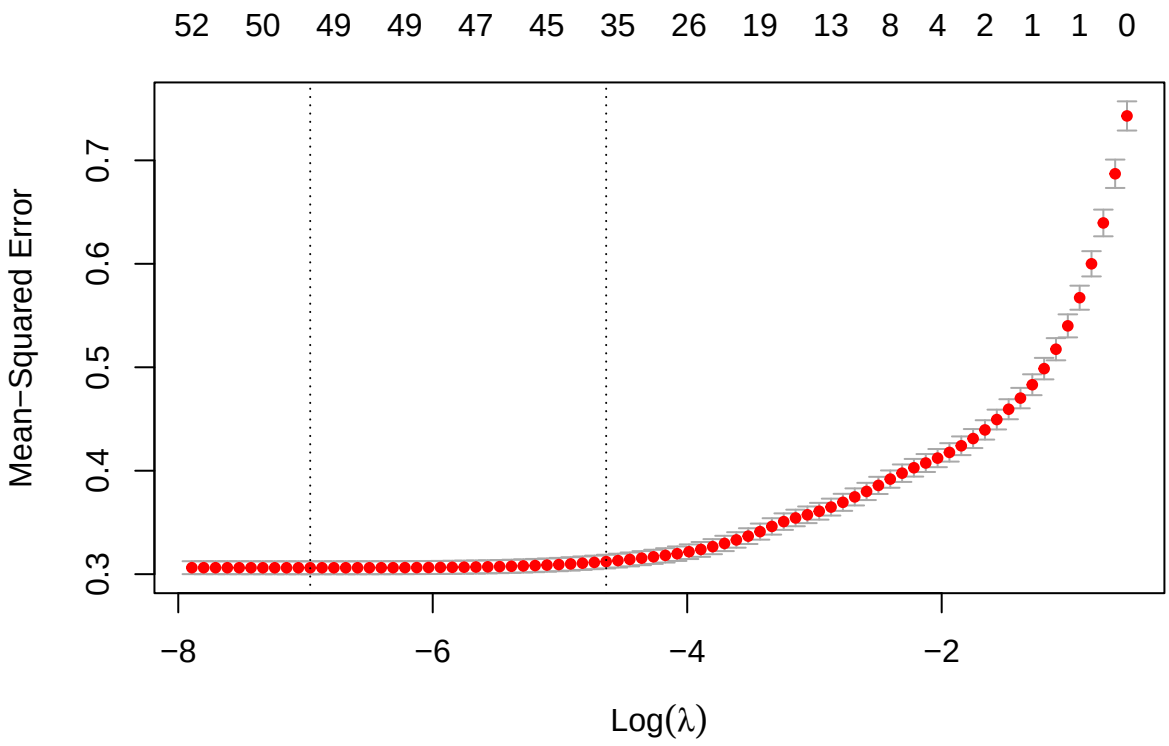
```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-4
```

```
set.seed(847)
X=model.matrix(score~.+log(members)+log(scored_by)+log(favorites)+
                 log(episode_duration)+log(total_duration)+0,data=dat2) #lasso
M=glmnet(X,dat2$score)
plot(M,xvar="lambda",label=TRUE)
```




```
CV=cv.glmnet(X,dat2$score)
plot(CV)
```




```
dgcm=coef(CV,s="lambda.1se")
length(dgcm)
```

```
## [1] 60
```

```
length(dgcm@i)
```

```
## [1] 38
```

```
coefs=dgcm[dgcm@i+1]
names(coefs)=rownames(dgcm)[dgcm@i+1]
coefs
```

```
##             (Intercept)               typemovie               typemusic
##            6.652705e-01            1.772628e-01            3.780879e-01
##                typeova             typespecial                 typetv
##           -1.450690e-02            2.041944e-01           -2.374300e-01
##                end_date      source4_koma_manga              sourcebook
##            1.881194e-05            4.038783e-02            6.985118e-02
##         sourcecard_game              sourcegame             sourcemanga
##           -9.320458e-03           -9.822210e-02            1.373703e-01
##       sourcemixed_media             sourcemusic             sourcenovel
##           -6.699516e-02           -6.261207e-02            1.355985e-01
##             sourceother         sourceweb_novel               favorites
##           -2.109790e-02            4.763217e-01            7.093451e-06
##        episode_duration                 ratingg                 ratingr
##            5.230863e-06            8.077148e-02           -9.384090e-02
##                ratingr+                 ratingrx                 sfwTrue
##           -3.614352e-01           -3.200029e-02            2.548271e-01
##              updated_at      start_seasonsummer          real_end_date
##            1.006409e-04           -1.291659e-03            2.310062e-07
##     broadcast_dayfriday     broadcast_daymonday   broadcast_daysaturday
```

```
##          -4.059373e-02          -9.710559e-02          -8.545983e-02
##    broadcast_daysunday  broadcast_daythursday   broadcast_daytuesday
##          -4.828334e-02          -1.386649e-01          -6.514805e-02
## broadcast_daywednesday           log(members)         log(scored_by)
##          -1.805311e-01           1.183344e-01           1.047403e-01
##  log(episode_duration)    log(total_duration)
##           3.050200e-02           1.636730e-01
```

Lasso regression was used to do model selection and dimensionality reduction for score vs 20 other variables (which include the variables used in the previous questions) plus log terms for some of them (particularly, continuous variables with a high left skewness). Out of 59 covariates, 37 were selected, as shown above, with the coefficient estimates.

7

To ensure reproducibility, I would make sure I set a seed before any code that involves using random variables. To ensure it is easily evaluated by others, I would minimize the use of magic numbers and dealing with special cases in the dataset. For example, in case a new column is added in an update that represents a categorical variable with too many classes for it to be used in the analysis, I would remove columns by having a threshold for max number of classes in a variable and then loop over the columns to filter the variables accordingly, instead of removing them manually.

8

An example of a possible ethical concern when using this dataset is when a model that predicts missing values in that dataset (e.x. through multiple imputation) is used; for example, it may wrongly predict an anime with unknown ratings to be “g” (i.e. safe) when the true rating is “r” (i.e. violence and profanity), increasing the probability of it being shown to children.